Advanced Encryption Standard

[Slide courtesy: Cryptography and network security by Bahrouz Fourozan]

AES

- Published by NIST (National Institute of Standards and Technology) in December 2001
  - First AES candidate conference
    - 15 out of 21 algorithms selected
  - Second AES candidate conference
    - 5 out of 15 selected as finalists
      - MARS, RC6, Rijndael, Serpent and Twofish
  - Third AES candidate conference
    - NIST announced Rijndael as the AES

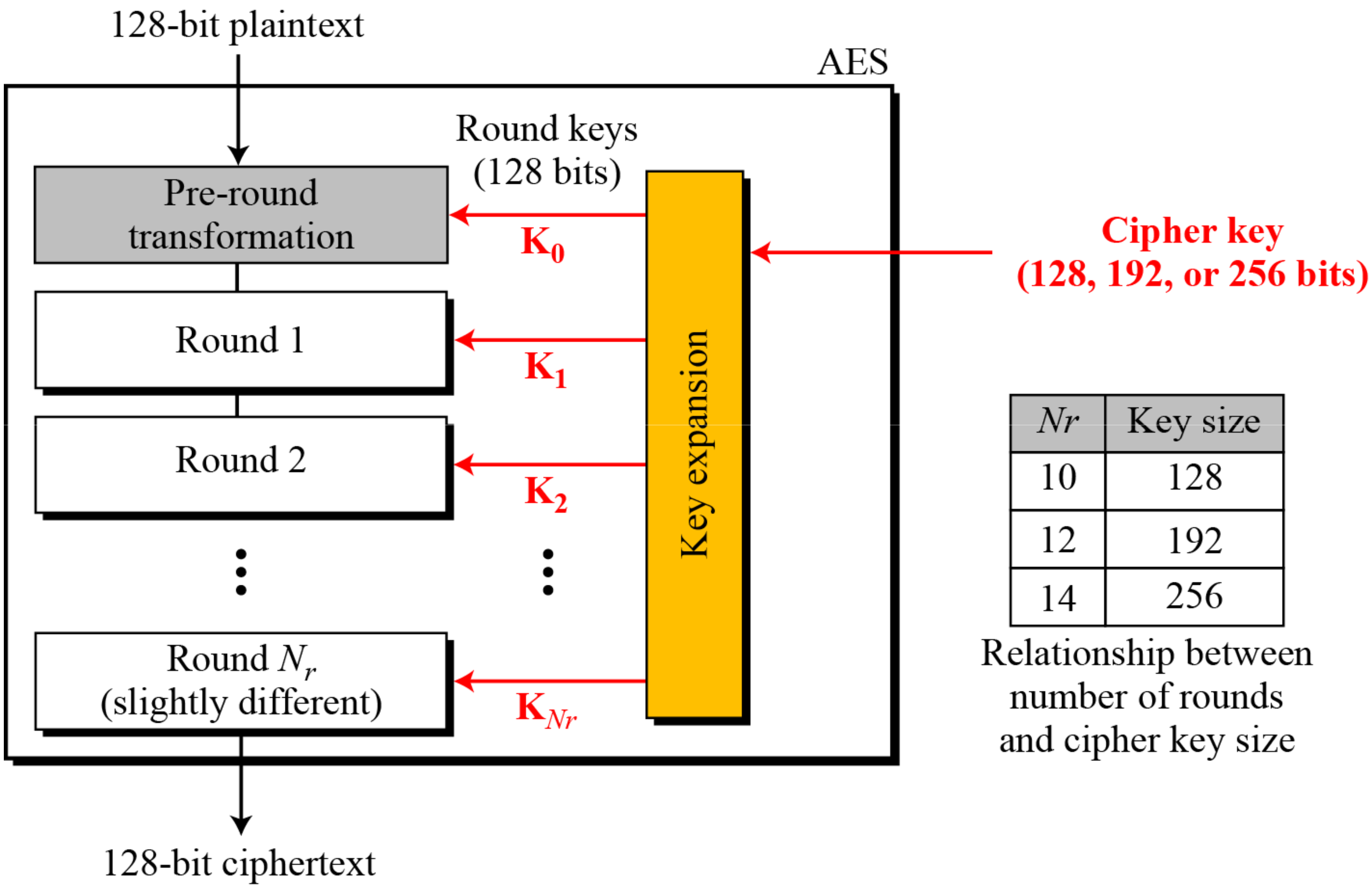
AES...

- The criteria defined by NIST for selecting AES fall into three areas:
  - Security
  - Cost
  - Implementation

AES…

- AES is a non-Feistel cipher that encrypts and decrypts a data block of 128 bits.
- It uses 10, 12, or 14 rounds.
- The key size, which can be 128, 192, or 256 bits, depends on the number of rounds.
  - But the round keys are always 128 bits

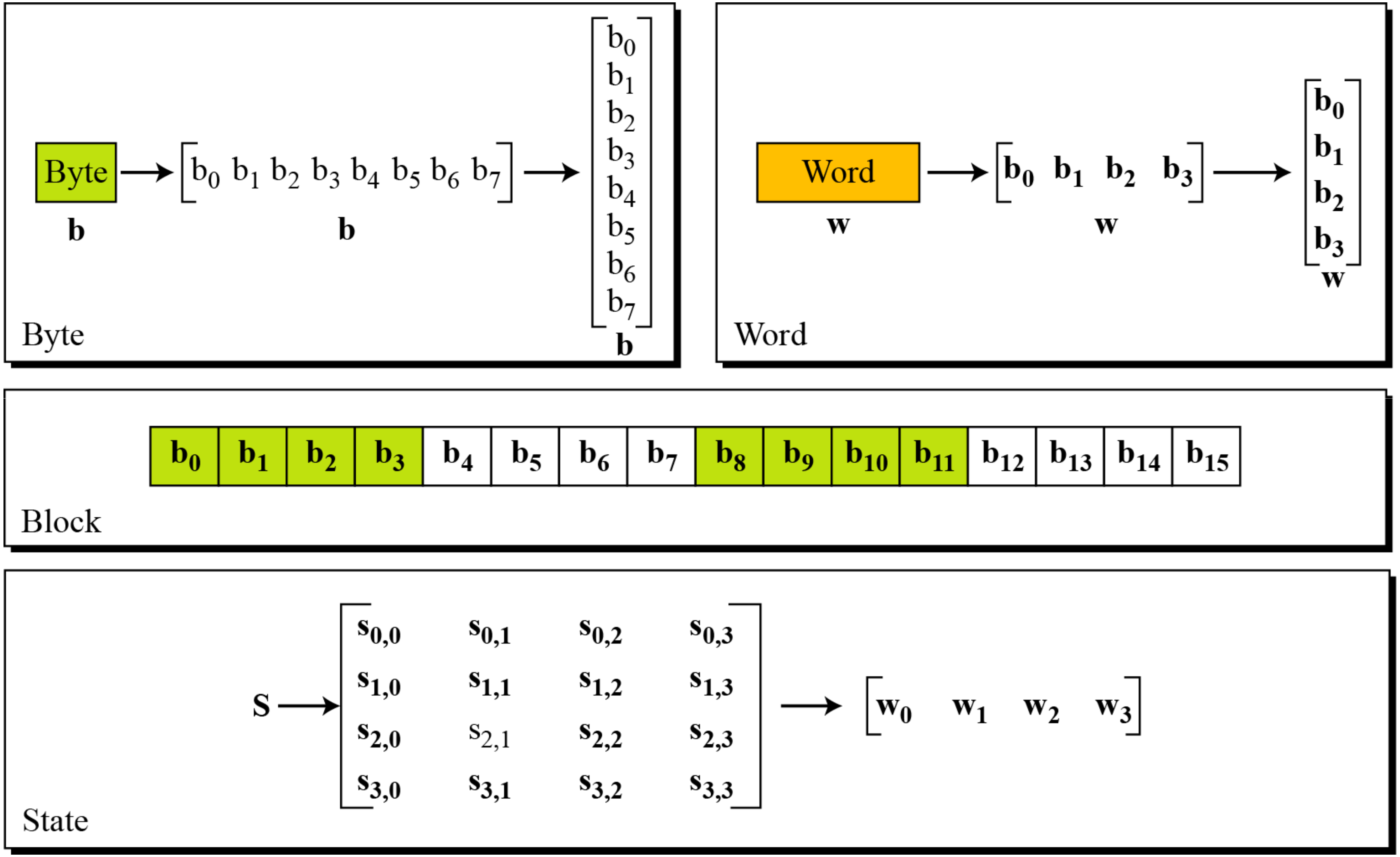
AES (General Design)…



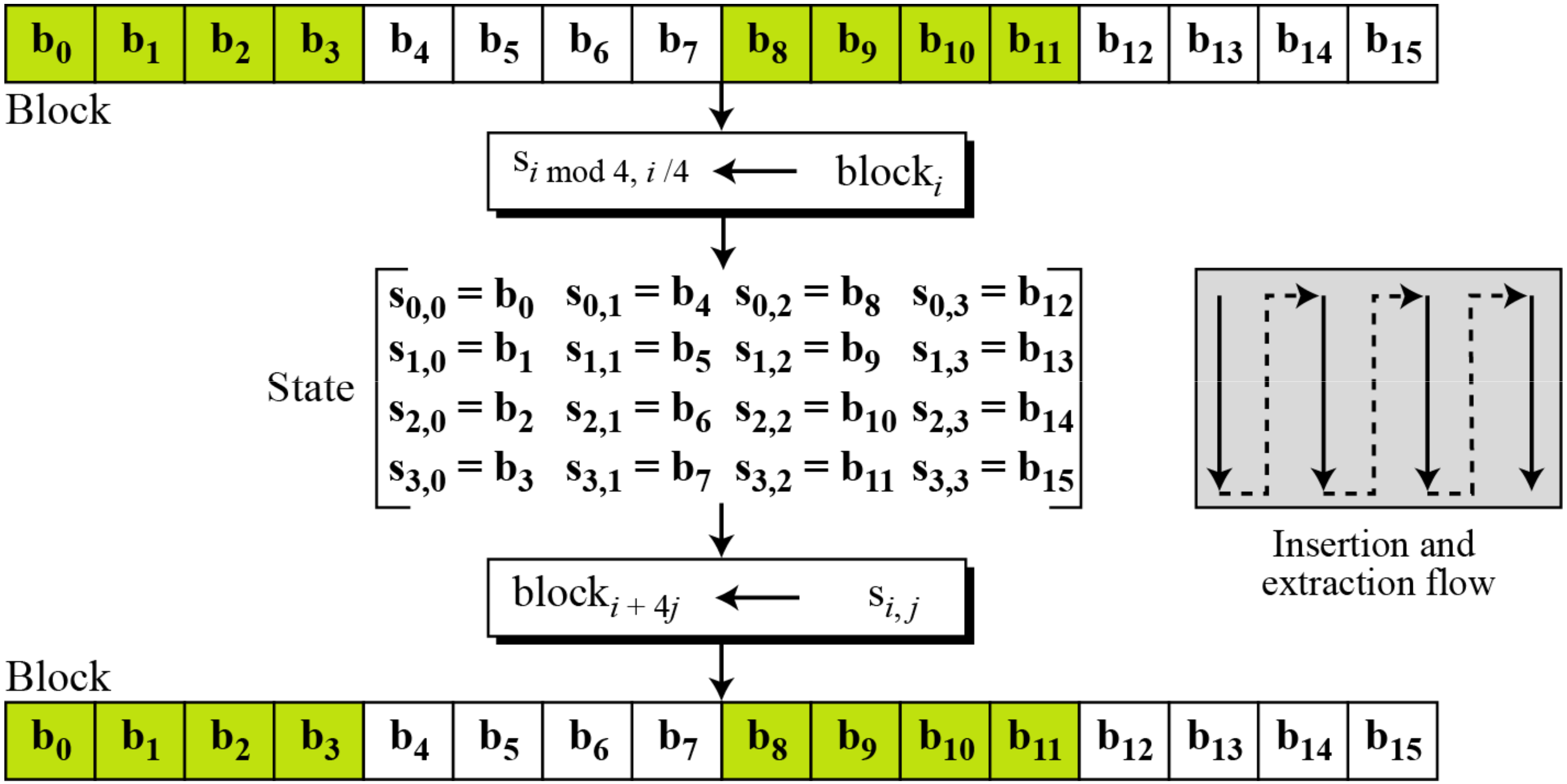
Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

AES (Data Units)...



AES (Data Units)...



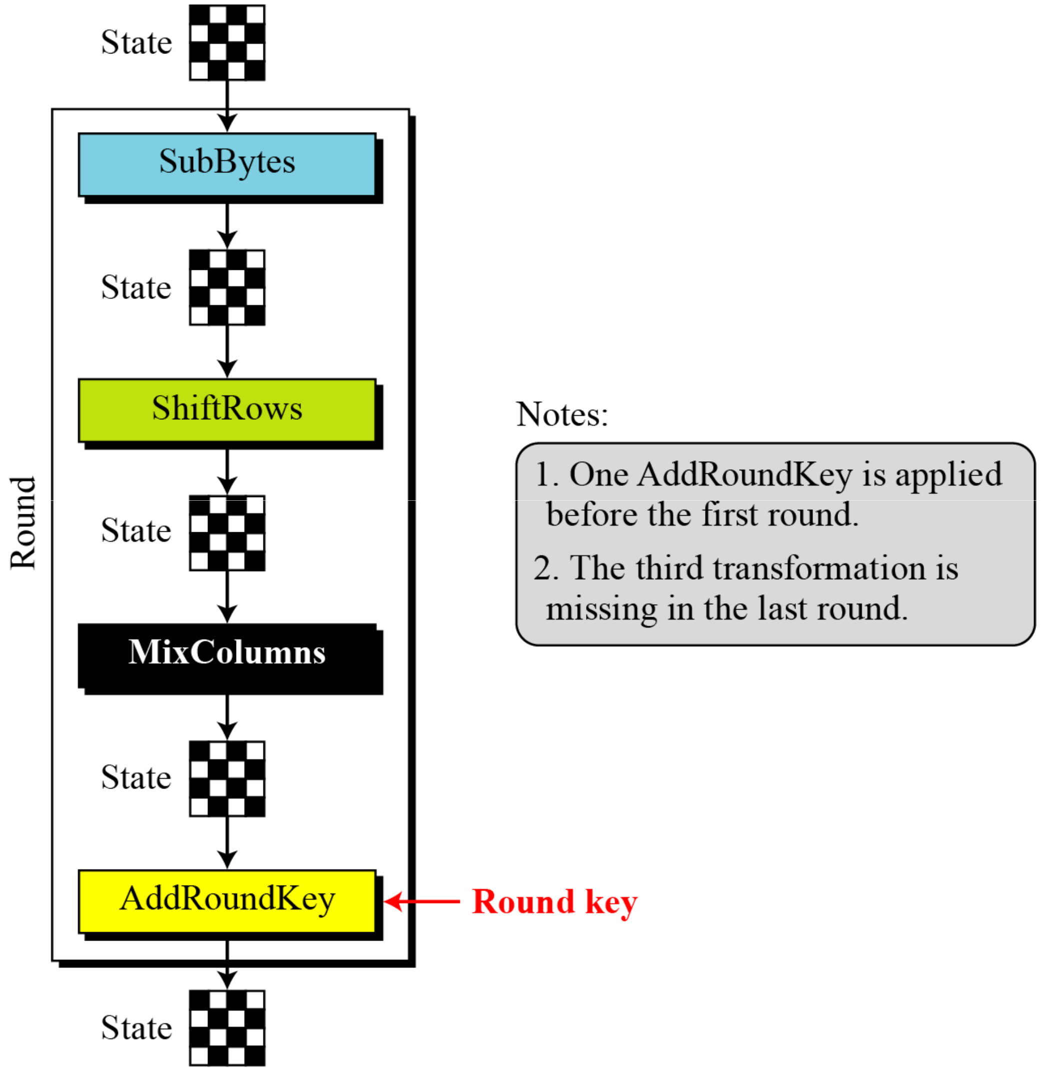
AES (Data Units)…

- Changing plaintext to states

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \text{State}$$

Structure of each round



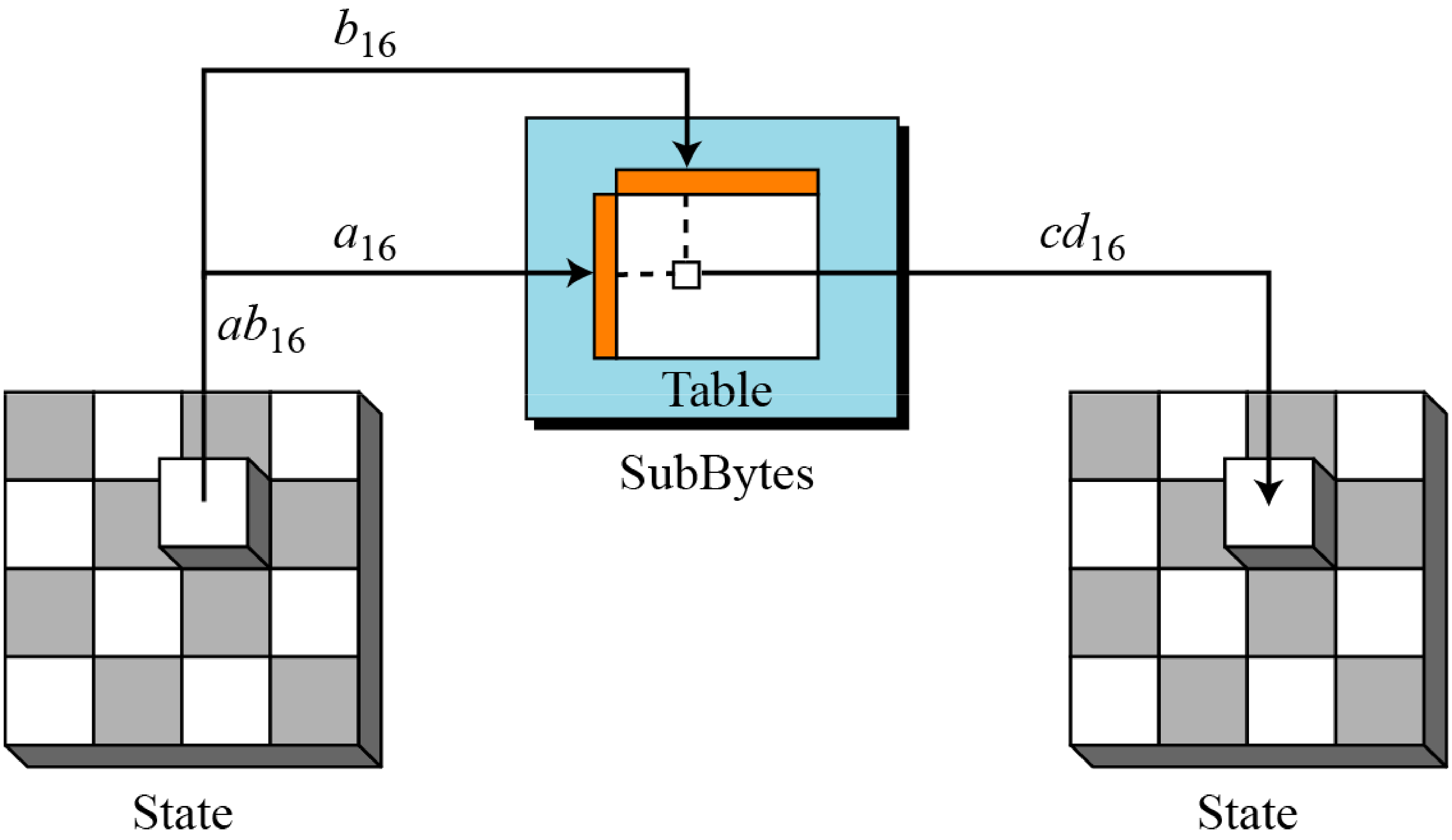
AES Transformations

- Four types
  - Substitution
  - Permutation
  - Mixing
  - Key-adding

Substitution

- AES, like DES, uses substitution. AES uses two invertible transformations.
- SubBytes
  - The first transformation, SubBytes, is used at the encryption site.
  - To substitute a byte, we interpret the byte as two hexadecimal digits.
  - The SubBytes operation involves 16 independent byte-to-byte transformations.

Substitution...



Substitution…

Table 7.1 *SubBytes transformation table*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8

Substitution…

Table 7.1 *SubBytes transformation table (continued)*

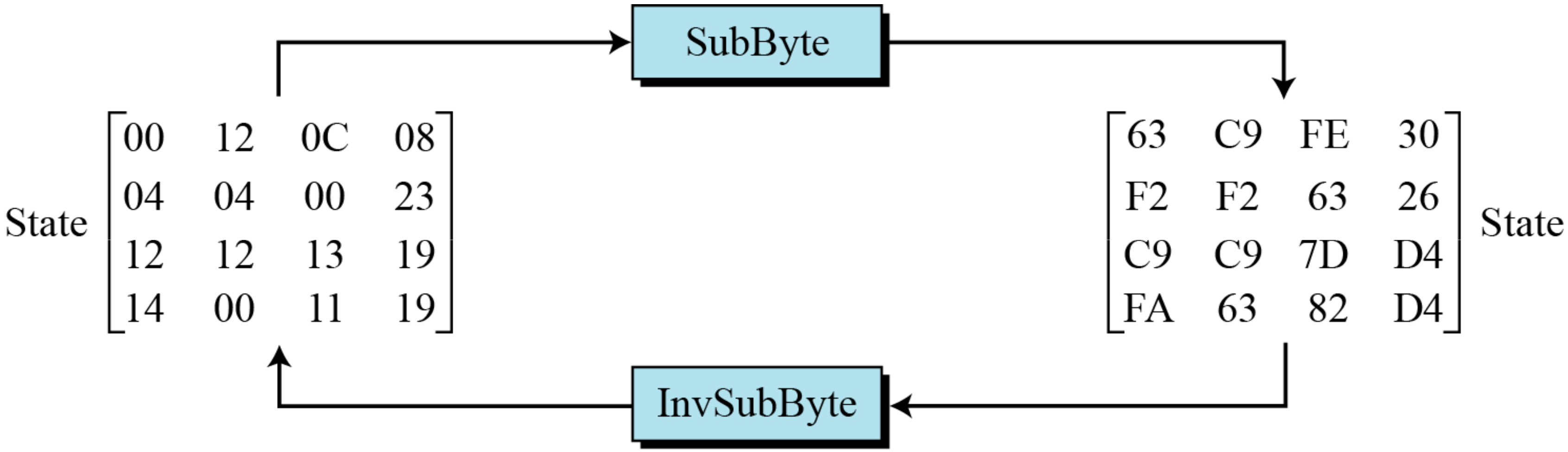
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	CB	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Substitution…

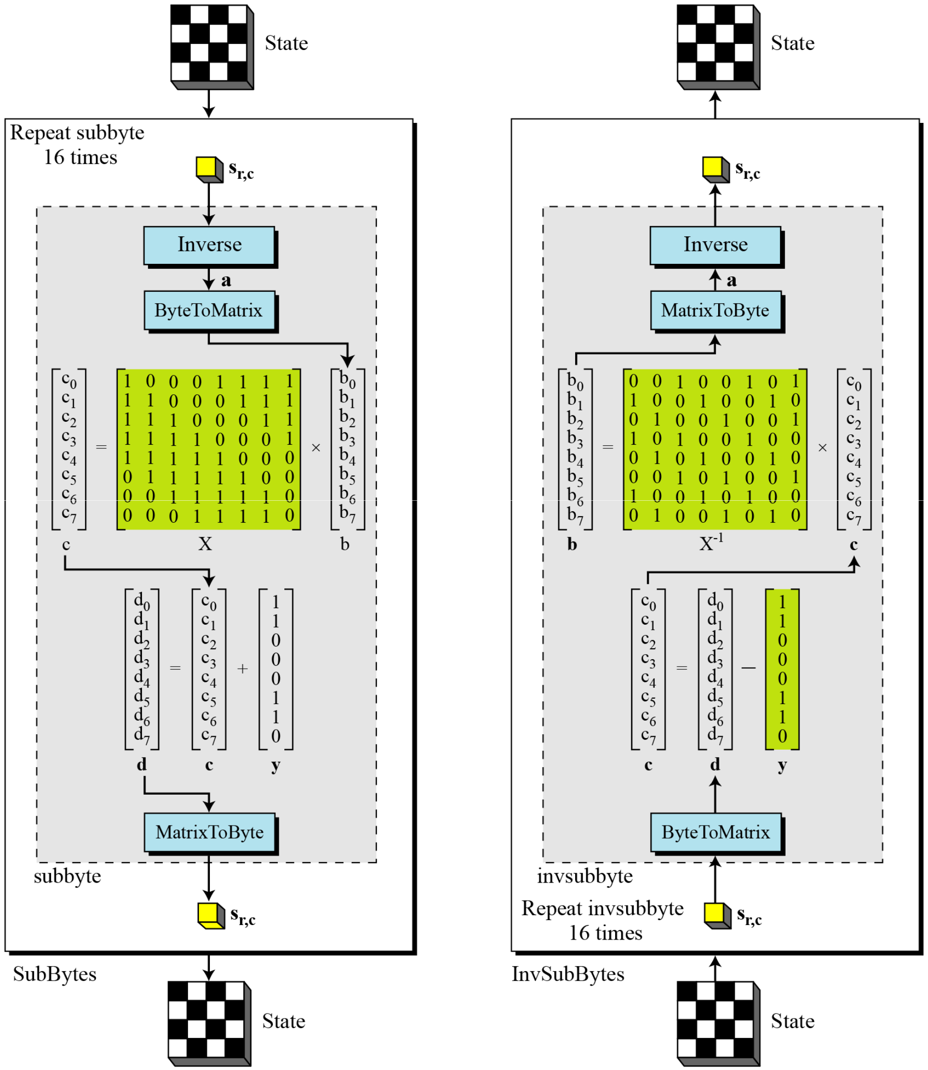
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Substitution…

- Example



SubBytes and InvSubBytes processes



SubBytes and InvSubBytes processes…

- Example
  - Let us show how the byte 0C is transformed to FE by subbyte routine and transformed back to 0C by the invsubbyte routine.

1. *subbyte:*
   a. The multiplicative inverse of 0C in GF(2^8) field is B0, which means **b** is (10110000).
   b. Multiplying matrix **X** by this matrix results in **c** = (10011101)
   c. The result of XOR operation is **d** = (11111110), which is FE in hexadecimal.

2. *invsubbyte:*
   a. The result of XOR operation is **c** = (10011101)
   b. The result of multiplying by matrix $\mathbf{X}^{-1}$ is (11010000) or B0
   c. The multiplicative inverse of B0 is 0C.

SubBytes and InvSubBytes processes…

Algorithm 7.1 *Pseudocode for SubBytes transformation*

```
SubBytes (S)
{
   for (r = 0 to 3)
     for (c = 0 to 3)
          S_{r,c} = subbyte (S_{r,c})
}

subbyte (byte)
{
   a ← byte^{-1}          // Multiplicative inverse in GF(2^8) with inverse of 00 to be 00
   ByteToMatrix (a, b)
   for (i = 0 to 7)
   {
      c_i ← b_i ⊕ b_{(i+4)mod 8} ⊕ b_{(i+5)mod 8} ⊕ b_{(i+6)mod 8} ⊕ b_{(i+7)mod 8}
      d_i ← c_i ⊕ ByteToMatrix (0x63)
   }
   MatrixToByte (d, d)
   byte ← d
}
```

Substitution…

- Exercise
  1. Can you prove formally that the subbyte and invsubbyte are inverses of each other?
  2. Can you prove that the subbyte transformation is nonlinear?

Substitution…

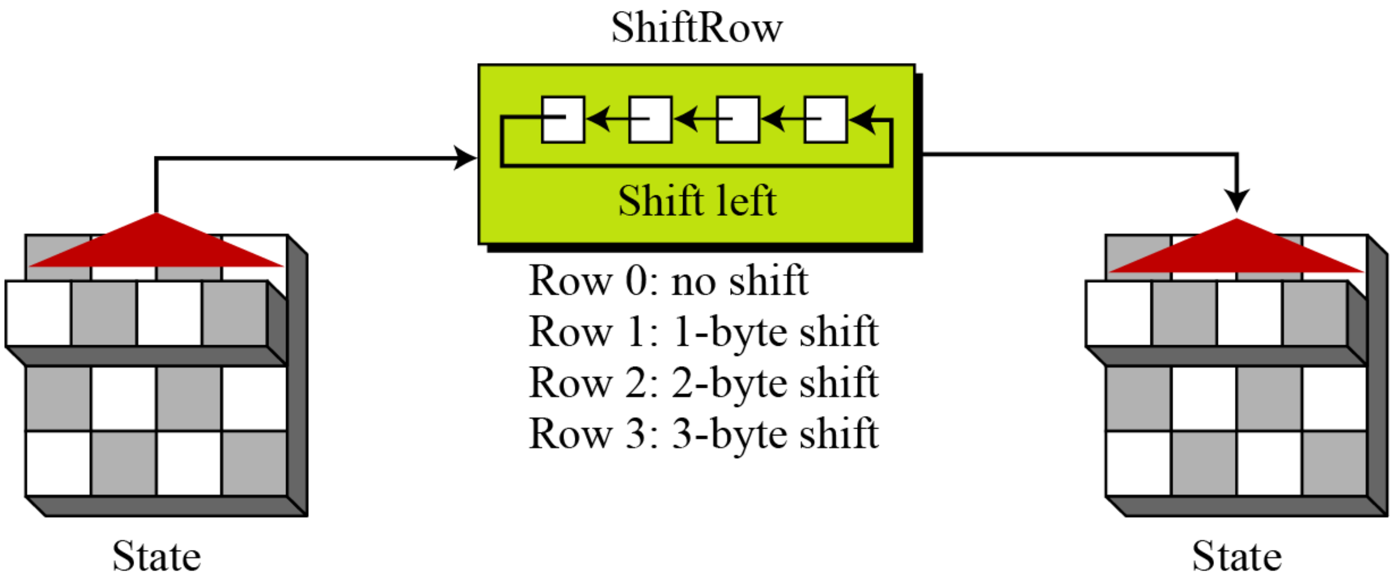
- Exercise- solution
  1. Can you prove formally that the subbyte and invsubbyte are inverses of each other?

$$\text{subbyte:} \quad \rightarrow \quad \mathbf{d} = \mathbf{X}\,(s_{r,c})^{-1} \oplus \mathbf{y}$$
$$\text{invsubbyte:} \quad \rightarrow \quad [\mathbf{X}^{-1}(\mathbf{d} \oplus \mathbf{y})]^{-1} = [\mathbf{X}^{-1}(\mathbf{X}\,(s_{r,c})^{-1} \oplus \mathbf{y} \;\oplus \mathbf{y})]^{-1} = [(s_{r,c})^{-1}]^{-1} = s_{r,c}$$

  2. It is the inverse operation, that makes the whole transformation nonlinear

Permutation

- ShiftRows
  - In the encryption, the transformation is called ShiftRows.



Permutation...

- InvShiftRows
  - In the decryption, the transformation is called InvShiftRows and the shifting is to the right.

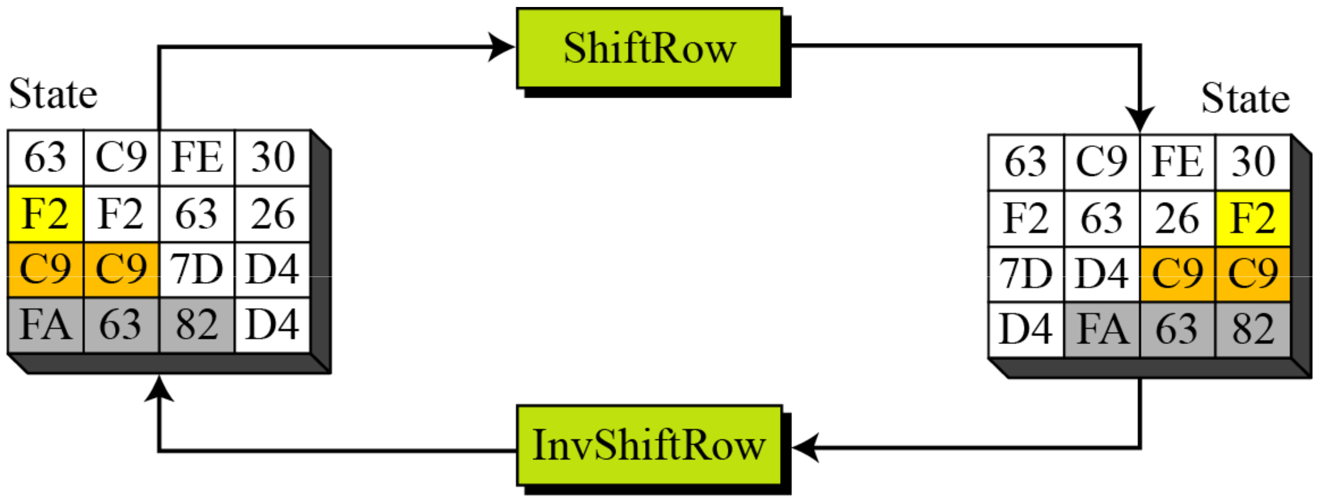
Algorithm 7.2 *Pseudocode for ShiftRows transformation*

```
ShiftRows (S)
{
    for (r = 1 to 3)
        shiftrow (s_r, r)                 // s_r is the rth row
}
```

```
shiftrow (row, n)                         // n is the number of bytes to be shifted
{
   CopyRow (row, t)                       // t is a temporary row
   for (c = 0 to 3)
        row_((c − n) mod 4) ← t_c
}
```

Permutation...

- Example



Mixing

- An interbyte transformation that changes the bits inside a byte, based on the bits inside the neighboring bytes.
  - We need to mix bytes to provide diffusion at the bit level.

$$\begin{matrix} a\mathbf{x} + b\mathbf{y} + c\mathbf{z} + d\mathbf{t} \\ e\mathbf{x} + f\mathbf{y} + g\mathbf{z} + h\mathbf{t} \\ i\mathbf{x} + j\mathbf{y} + k\mathbf{z} + l\mathbf{t} \\ m\mathbf{x} + n\mathbf{y} + o\mathbf{z} + p\mathbf{t} \end{matrix} \begin{matrix} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{matrix} \begin{bmatrix} \square \\ \square \\ \square \\ \square \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \times \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \\ \mathbf{t} \end{bmatrix}$$

New matrix | **Constant matrix** | Old matrix

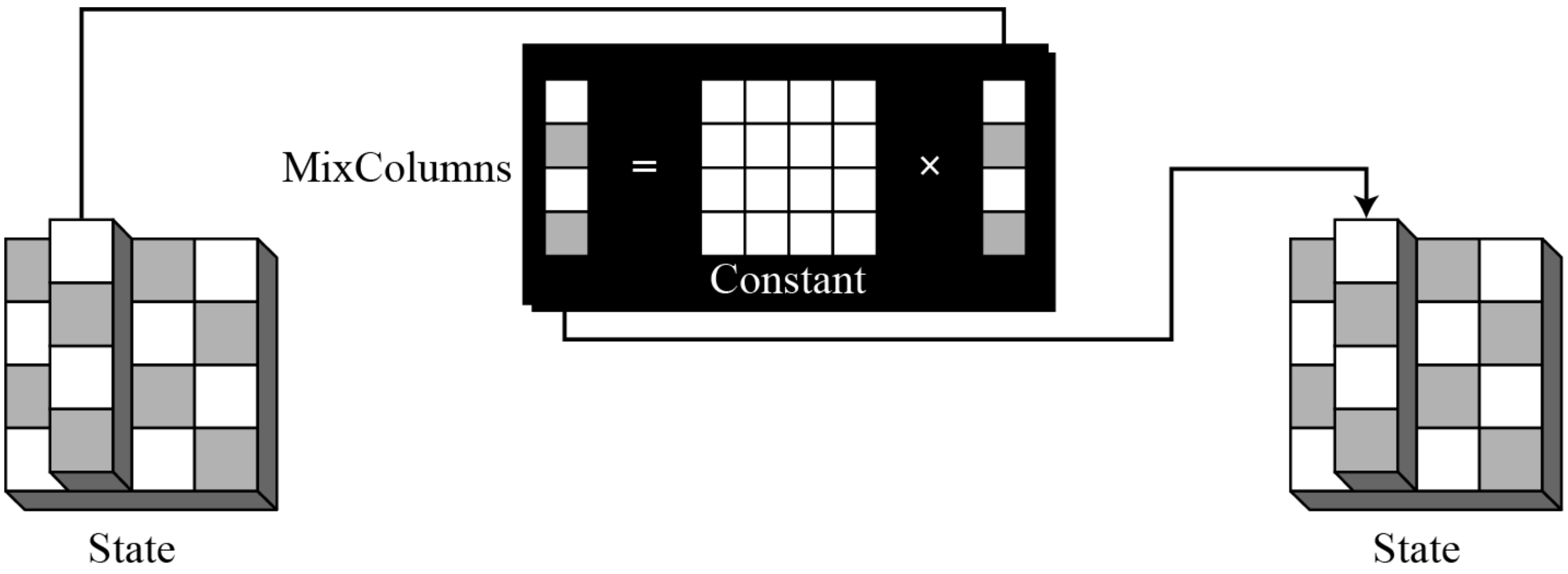
Mixing...

- Constant matrices used by MixColumns and InvMixColumns

$$\underbrace{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}}_{\mathrm{C}} \xleftrightarrow{\text{Inverse}} \underbrace{\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}}_{\mathrm{C}^{-1}}$$

Mixing...

- MixColumns
  - operates at the column level
  - transforms each column of the state to a new column.



Mixing…

- InvMixColumns
  - basically the same as the MixColumns transformation but the inverse

Algorithm 7.3 *Pseudocode for MixColumns transformation*

```
MixColumns (S)
{
     for (c = 0 to 3)
          mixcolumn (s_c)
}
mixcolumn (col)
{
   CopyColumn (col, t)                         // t is a temporary column
```

$$\mathbf{col}_0 \leftarrow (0x02) \bullet \mathbf{t}_0 \oplus (0x03 \bullet \mathbf{t}_1) \oplus \mathbf{t}_2 \oplus \mathbf{t}_3$$

$$\mathbf{col}_1 \leftarrow \mathbf{t}_0 \oplus (0x02) \bullet \mathbf{t}_1 \oplus (0x03) \bullet \mathbf{t}_2 \oplus \mathbf{t}_3$$

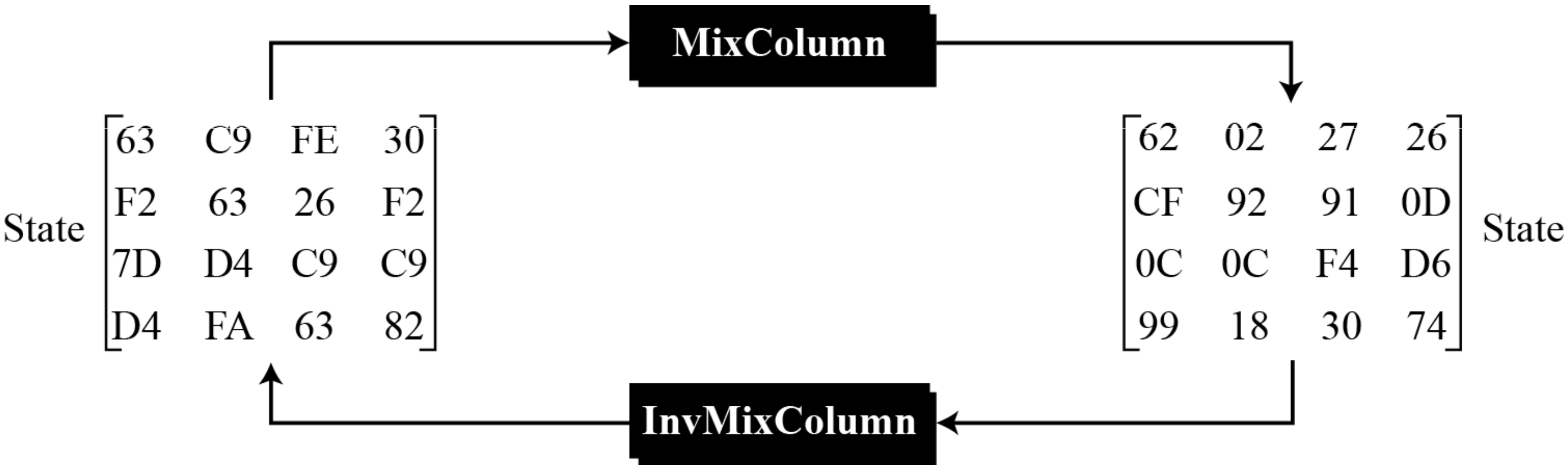
$$\mathbf{col}_2 \leftarrow \mathbf{t}_0 \oplus \mathbf{t}_1 \oplus (0x02) \bullet \mathbf{t}_2 \oplus (0x03) \bullet \mathbf{t}_3$$

$$\mathbf{col}_3 \leftarrow (0x03 \bullet \mathbf{t}_0) \oplus \mathbf{t}_1 \oplus \mathbf{t}_2 \oplus (0x02) \bullet \mathbf{t}_3$$

```
}
```

Mixing...

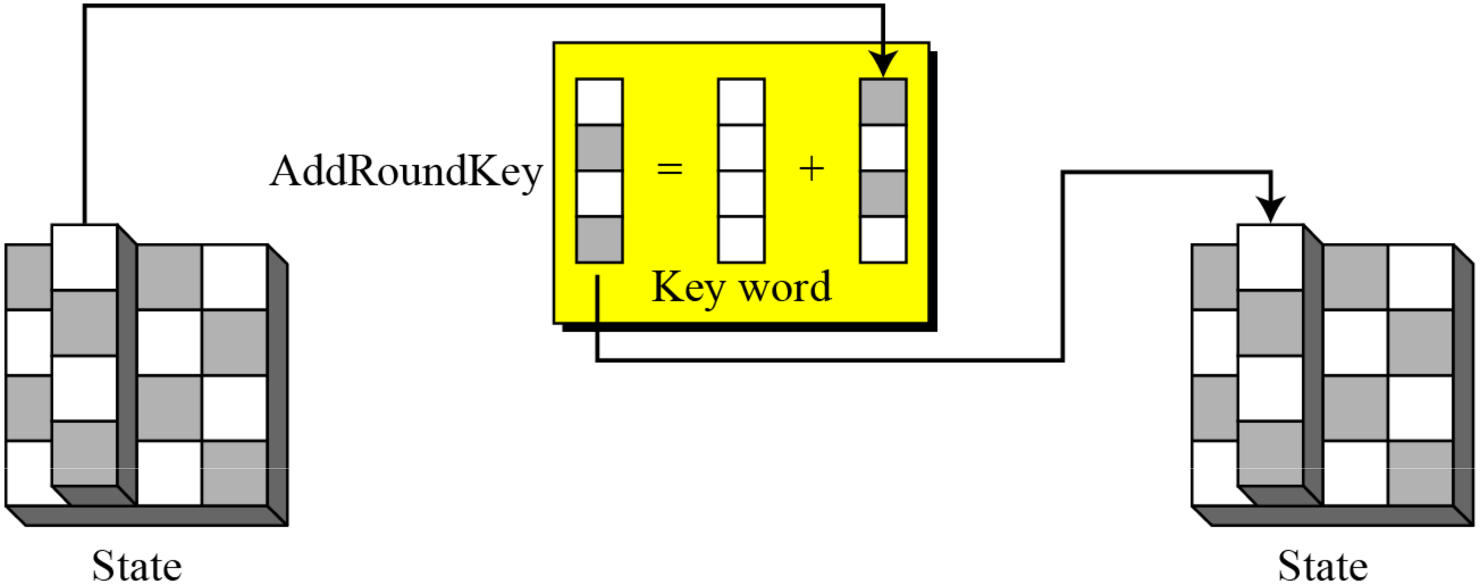
- Example
  - how a state is transformed using the MixColumns transformation. The figure also shows that the InvMixColumns transformation creates the original one.



Key Adding

- AddRoundKey
  - AddRoundKey proceeds one column at a time.
  - AddRoundKey adds a round key word with each state column matrix
  - the operation in AddRoundKey is matrix addition.

Key Adding…



Algorithm 7.4 *Pseudocode for AddRoundKey transformation*

```
AddRoundKey (S)
{
    for (c = 0 to 3)
        s_c ← s_c ⊕ w_round + 4c
}
```

Key Expansion

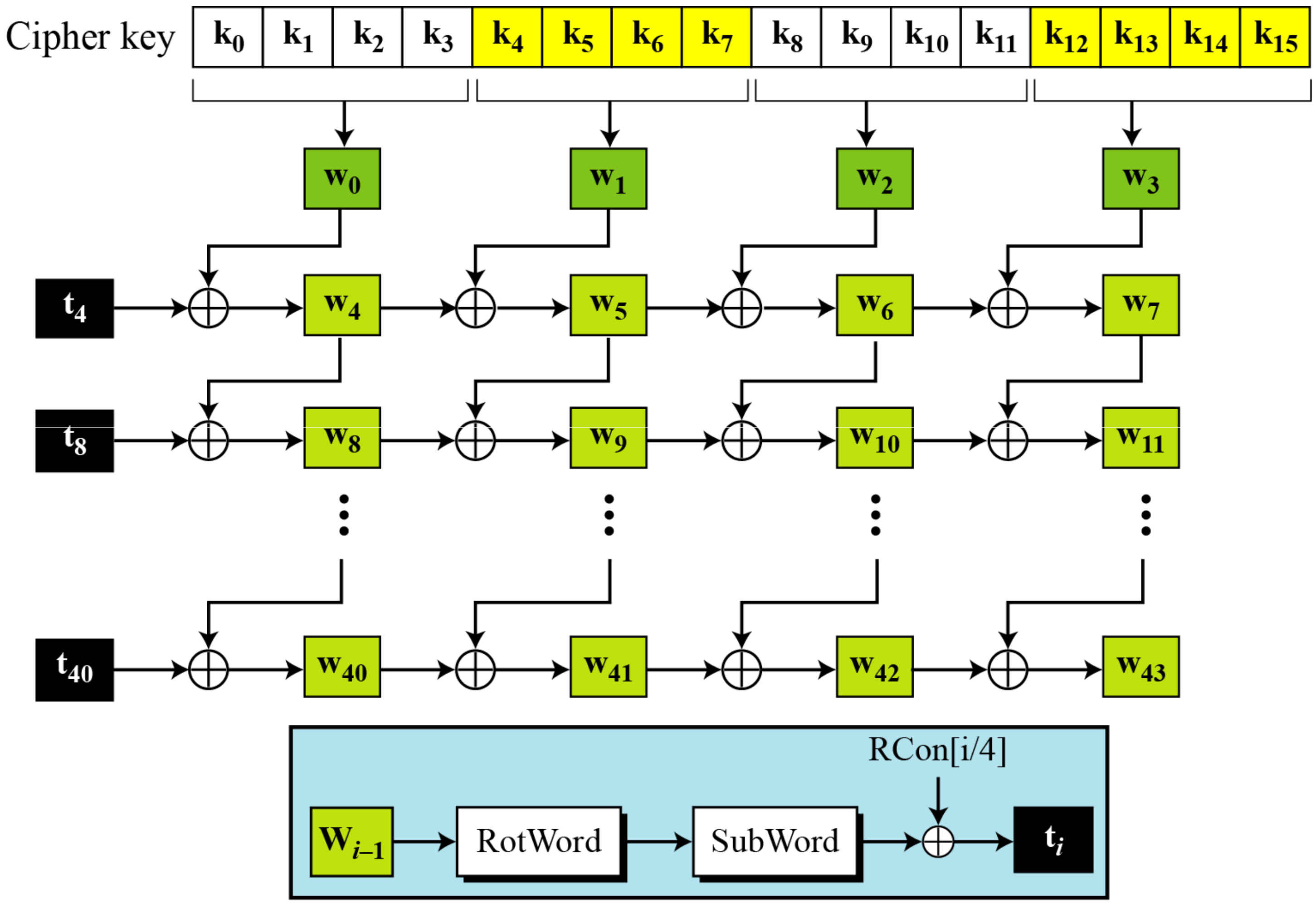
- To create round keys for each round, AES uses a key-expansion process.
- If the number of rounds is Nr , the key-expansion routine creates Nr + 1 128-bit round keys from one single 128-bit cipher key.

Key Expansion...

Table 7.3 *Words for each round*

Round	*Words*			
Pre-round	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$
1	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$	$\mathbf{w}_7$
2	$\mathbf{w}_8$	$\mathbf{w}_9$	$\mathbf{w}_{10}$	$\mathbf{w}_{11}$
. . .	. . .			
N_r	$\mathbf{w}_{4N_r}$	$\mathbf{w}_{4N_r+1}$	$\mathbf{w}_{4N_r+2}$	$\mathbf{w}_{4N_r+3}$

Key Expansion in AES-128...



Making of t_i (temporary) words $i = 4\ N_r$

Key Expansion in AES-128...

Table 7.4 *RCon constants*

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	$(\underline{\mathbf{01}}\ 00\ 00\ 00)_{16}$	6	$(\underline{\mathbf{20}}\ 00\ 00\ 00)_{16}$
2	$(\underline{\mathbf{02}}\ 00\ 00\ 00)_{16}$	7	$(\underline{\mathbf{40}}\ 00\ 00\ 00)_{16}$
3	$(\underline{\mathbf{04}}\ 00\ 00\ 00)_{16}$	8	$(\underline{\mathbf{80}}\ 00\ 00\ 00)_{16}$
4	$(\underline{\mathbf{08}}\ 00\ 00\ 00)_{16}$	9	$(\underline{\mathbf{1B}}\ 00\ 00\ 00)_{16}$
5	$(\underline{\mathbf{10}}\ 00\ 00\ 00)_{16}$	10	$(\underline{\mathbf{36}}\ 00\ 00\ 00)_{16}$

Key Expansion in AES-128...

- The key-expansion routine can either use the above table when calculating the words or use the GF(2^8) field to calculate the leftmost byte dynamically, as shown below (prime is the irreducible polynomial):

$$
\begin{array}{llllll}
RC_1 & \rightarrow x^{1-1} = x^0 & \bmod\ prime & = 1 & \rightarrow 00000001 & \rightarrow 01_{16} \\
RC_2 & \rightarrow x^{2-1} = x^1 & \bmod\ prime & = x & \rightarrow 00000010 & \rightarrow 02_{16} \\
RC_3 & \rightarrow x^{3-1} = x^2 & \bmod\ prime & = x^2 & \rightarrow 00000100 & \rightarrow 04_{16} \\
RC_4 & \rightarrow x^{4-1} = x^3 & \bmod\ prime & = x^3 & \rightarrow 00001000 & \rightarrow 08_{16} \\
RC_5 & \rightarrow x^{5-1} = x^4 & \bmod\ prime & = x^4 & \rightarrow 00010000 & \rightarrow 10_{16} \\
RC_6 & \rightarrow x^{6-1} = x^5 & \bmod\ prime & = x^5 & \rightarrow 00100000 & \rightarrow 20_{16} \\
RC_7 & \rightarrow x^{7-1} = x^6 & \bmod\ prime & = x^6 & \rightarrow 01000000 & \rightarrow 40_{16} \\
RC_8 & \rightarrow x^{8-1} = x^7 & \bmod\ prime & = x^7 & \rightarrow 10000000 & \rightarrow 80_{16} \\
RC_9 & \rightarrow x^{9-1} = x^8 & \bmod\ prime & = x^4 + x^3 + x + 1 & \rightarrow 00011011 & \rightarrow 1B_{16} \\
RC_{10} & \rightarrow x^{10-1} = x^9 & \bmod\ prime & = x^5 + x^4 + x^2 + x & \rightarrow 00110110 & \rightarrow 36_{16}
\end{array}
$$

Key Expansion in AES-128...

- An illustration
  - how the keys for each round are calculated assuming that the 128-bit cipher key agreed upon by Alice and Bob is (24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87)$_{16}$.

Table 7.5 *Key expansion example*

Round	*Values of t's*	*First word in the round*	*Second word in the round*	*Third word in the round*	*Fourth word in the round*
—		w_{00} = 2475A2B3	w_{01} = 34755688	w_{02} = 31E21200	w_{03} = 13AA5487
1	AD20177D	w_{04} = 8955B5CE	w_{05} = BD20E346	w_{06} = 8CC2F146	w_{07} = 9F68A5C1
2	470678DB	w_{08} = CE53CD15	w_{09} = 73732E53	w_{10} = FFB1DF15	w_{11} = 60D97AD4
3	31DA48D0	w_{12} = FF8985C5	w_{13} = 8CFAAB96	w_{14} = 734B7483	w_{15} = 2475A2B3
4	47AB5B7D	w_{16} = B822deb8	w_{17} = 34D8752E	w_{18} = 479301AD	w_{19} = 54010FFA
5	6C762D20	w_{20} = D454F398	w_{21} = E08C86B6	w_{22} = A71F871B	w_{23} = F31E88E1
6	52C4F80D	w_{24} = 86900B95	w_{25} = 661C8D23	w_{26} = C1030A38	w_{27} = 321D82D9
7	E4133523	w_{28} = 62833EB6	w_{29} = 049FB395	w_{30} = C59CB9AD	w_{31} = F7813B74
8	8CE29268	w_{32} = EE61ACDE	w_{33} = EAFE1F4B	w_{34} = 2F62A6E6	w_{35} = D8E39D92
9	0A5E4F61	w_{36} = E43FE3BF	w_{37} = 0EC1FCF4	w_{38} = 21A35A12	w_{39} = F940C780
10	3FC6CD99	w_{40} = DBF92E26	w_{41} = D538D2D2	w_{42} = F49B88C0	w_{43} = 0DDB4F40

Key Expansion in AES-128...

- Each round key in AES depends on the previous round key.
- The dependency, however, is nonlinear because of SubWord transformation.
- The addition of the round constants also guarantees that each round key will be different from the previous one.
- An illustration

Cipher Key 1: 12 45 A2 A1 23 31 A4 A3 B2 CC A**A** 34 C2 BB 77 23
Cipher Key 2: 12 45 A2 A1 23 31 A4 A3 B2 CC A**B** 34 C2 BB 77 23

Key Expansion in AES-128...

Table 7.6 *Comparing two sets of round keys*

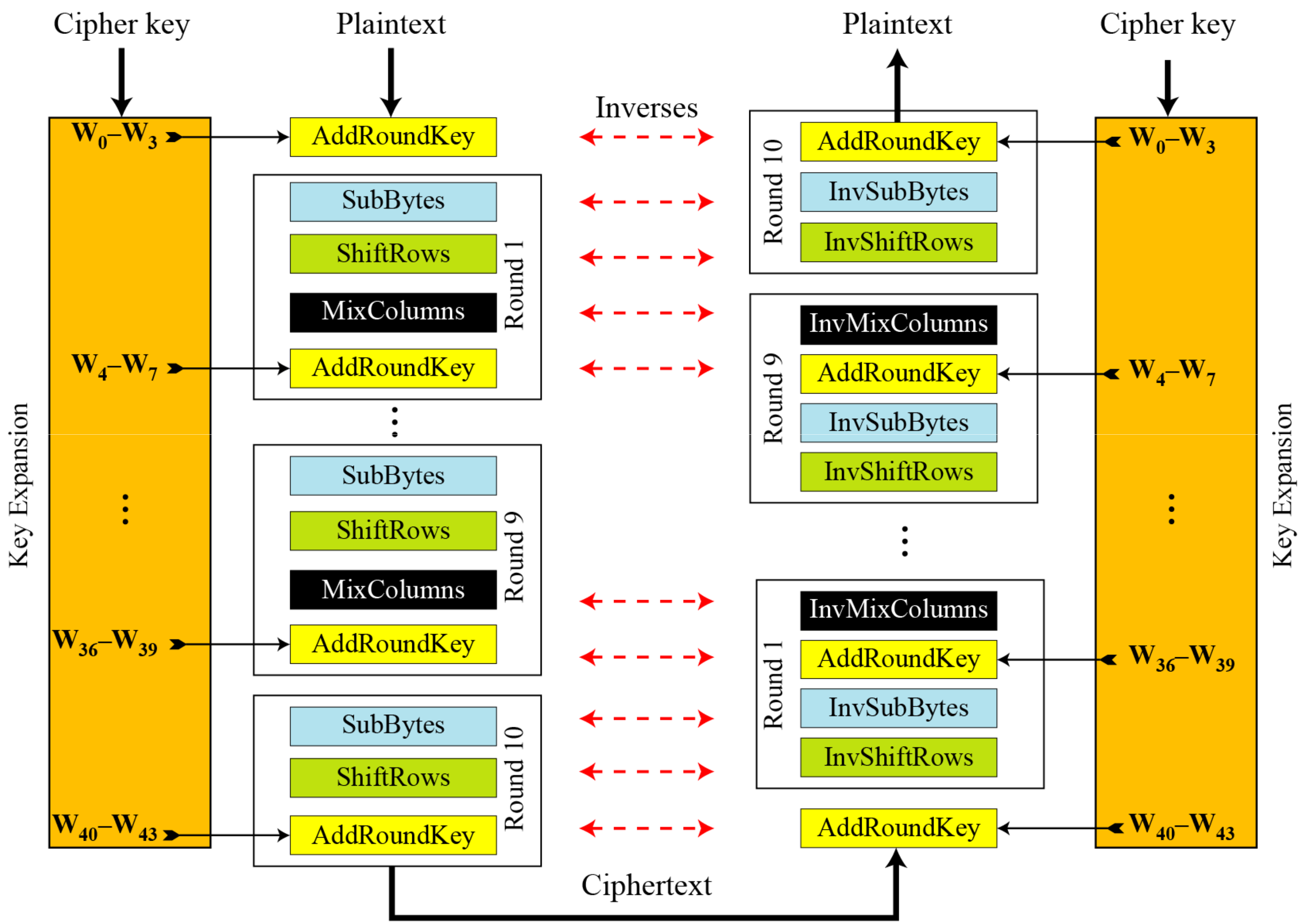
R.	*Round keys for set 1*	*Round keys for set 2*	*B. D.*
—	1245A2A1 2331A4A3 B2CCAA34 C2BB7723	1245A2A1 2331A4A3 B2CCAB34 C2BB7723	01
1	F9B08484 DA812027 684D8A13 AAF6FD30	F9B08484 DA812027 684D8B13 AAF6FC30	02
2	B9E48028 6365A00F 0B282A1C A1DED72C	B9008028 6381A00F 0BCC2B1C A13AD72C	17
3	A0EAF11A C38F5115 C8A77B09 6979AC25	3D0EF11A 5E8F5115 55437A09 F479AD25	30
4	1E7BCEE3 DDF49FF6 1553E4FF 7C2A48DA	839BCEA5 DD149FB0 8857E5B9 7C2E489C	31
5	EB2999F3 36DD0605 238EE2FA 5FA4AA20	A2C910B5 7FDD8F05 F78A6ABC 8BA42220	34
6	82852E3C B4582839 97D6CAC3 C87260E3	CB5AA788 B487288D 430D4231 C8A96011	56
7	82553FD4 360D17ED A1DBDD2E 69A9BDCD	588A2560 EC0D0DED AF004FDC 67A92FCD	50
8	D12F822D E72295C0 46F948EE 2F50F523	0B9F98E5 E7929508 4892DAD4 2F3BF519	44
9	99C9A438 7EEB31F8 38127916 17428C35	F2794CF0 15EBD9F8 5D79032C 7242F635	51
10	83AD32C8 FD460330 C5547A26 D216F613	E83BDAB0 FDD00348 A0A90064 D2EBF651	52

Key Expansion in AES-128...

- What about weak keys for AES???

Pre-round:	00000000	00000000	00000000	00000000
Round 01:	62636363	62636363	62636363	62636363
Round 02:	9B9898C9	F9FBFBAA	9B9898C9	F9FBFBAA
Round 03:	90973450	696CCFFA	F2F45733	0B0FAC99
...	...	...	...	...
Round 10:	B4EF5BCB	3E92E211	23E951CF	6F8F188E

The cipher



Cipher key
Plaintext
Plaintext
Cipher key
Inverses
W_0–W_3
AddRoundKey
SubBytes
ShiftRows
MixColumns
AddRoundKey
Round 1
W_4–W_7
Key Expansion
SubBytes
ShiftRows
MixColumns
AddRoundKey
Round 9
W_{36}–W_{39}
SubBytes
ShiftRows
AddRoundKey
Round 10
W_{40}–W_{43}
Round 10
AddRoundKey
InvSubBytes
InvShiftRows
Round 9
InvMixColumns
AddRoundKey
InvSubBytes
InvShiftRows
Round 1
InvMixColumns
AddRoundKey
InvSubBytes
InvShiftRows
AddRoundKey
W_0–W_3
W_4–W_7
W_{36}–W_{39}
W_{40}–W_{43}
Key Expansion
Ciphertext

Analysis

- The result of encryption when the plaintext is made of all 0s.

Plaintext:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Cipher Key:	24	75	A2	B3	34	75	56	88	31	E2	12	00	13	AA	54	87
Ciphertext:	63	2C	D4	5E	5D	56	ED	B5	62	04	01	A0	AA	9C	2D	8D

Analysis…

- The avalanche effect.

Plaintext 1:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Plaintext 2:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01
Ciphertext 1:	63	2C	D4	5E	5D	56	ED	B5	62	04	01	A0	AA	9C	2D	8D
Ciphertext 2:	26	F3	9B	BC	A1	9C	0F	B7	C7	2E	7E	30	63	92	73	13

Analysis...

- The effect of using a cipher key in which all bits are 0s.

Plaintext:	00	04	12	14	12	04	12	00	0c	00	13	11	08	23	19	19
Cipher Key:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Ciphertext:	5A	6F	4B	67	57	B7	A5	D2	C4	30	91	ED	64	9A	42	72